

Amendments to the Claims

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Presently Amended) A computer system that employs a plurality of threads of execution to perform a parallel-execution operation in which the threads identify tasks dynamically and in which the computer system comprises:
 - A) ~~provides a global status word that includes a mechanism that~~ associates a separate status-word field ~~associated~~ with each of the threads; and
 - B) ~~so a mechanism that~~ operates the threads in a manner that each thread:
 - i) executes a task-finding routine to find tasks previously identified dynamically and performs tasks thereby found, with ~~the~~ its associated status-word field ~~associated with that thread~~ containing an ~~activity representing a value~~ indicating it is active, until the task-finding routine finds no more tasks;
 - ii) when the task-finding routine finds no more tasks, sets the contents of ~~the~~ its associated status-word field ~~associated with that thread~~ to an ~~inactivity indicating a~~ value indicating it is inactive;
 - iii) while the status-word field associated with any other thread contains an ~~activity indicating a~~ value indicating that the other thread is active, searches for a task ~~and, and,~~ if it finds one, sets ~~the~~ its associated status-word field contents to ~~the activity indicating a~~ value indicating that it is active before attempting to execute a task; and
 - iv) if none of the status-word fields associated with other threads contains an ~~activity indicating a~~ value indicating that an associated thread is active, terminates its performance of the parallel-execution operation.

- 1 2. (Original) A computer system as defined in claim 1 wherein the parallel-execution
2 operation is a garbage-collection operation.
- 1 3. (Original) A computer system as defined in claim 1 wherein:
2 A) each thread has associated with it a respective work queue in
3 which it places task identifiers of tasks that identifies dynamically;
4 B) the task-finding routine executed by an executing thread includes
5 performing an initial search for a task identifiers in the work queue associated
6 with the executing thread and, if that work queue contains no task identifiers that
7 the executing thread can claim, thereafter performing a further search for a task
8 identifier in at least one other task-storage location.
- 1 4. (Original) A computer system as defined in claim 3 wherein the parallel-execution
2 operation is a garbage-collection operation.
- 1 5. (Original) A computer system as defined in claim 3 wherein the at least one other
2 task-storage location includes at least one work queue associated with a thread
3 other than the executing thread.
- 1 6. (Original) A computer system as defined in claim 5 wherein:
2 A) there is a size limit associated with each work queue;
3 B) when a given thread dynamically identifies a given task that would
4 cause the number of task entries in the work queue associated with the given
5 thread to exceed the size limit if a task identifier that identifies it were placed in
6 that work queue, the given thread instead places that task identifier in an
7 overflow list instead of in that work queue; and
8 C) the at least one other task-storage location includes at least one
9 such over flow list.

- 1 7. (Original) A computer system as defined in claim 5 wherein the task-finding
2 routine includes selecting in a random manner the at least one work queue
3 associated with a thread other than the executing thread.
- 1 8. (Original) A computer system as defined in claim 5 wherein the further search
2 includes repeatedly searching a work queue associated with a thread other than
3 the executing thread until the executing thread thereby finds a task or has
4 performed a number of repetitions equal to a repetition limit greater than one.
- 1 9. (Original) A computer system as defined in claim 8 wherein the task-finding
2 routine includes selecting in a random manner the at least one work queue
3 associated with a thread other than the executing thread.
- 1 10. (Original) A computer system as defined in claim 3 wherein:
2 A) there is a size limit associated with each work queue;
3 B) when a given thread dynamically identifies a given task that would
4 cause the number of task entries in the work queue associated with the given
5 thread to exceed the size limit if a task identifier that identifies it were placed in
6 that work queue, the given thread instead places that task identifier in an
7 overflow list instead of in that work queue; and
8 C) the at least one other task-storage location includes at least one
9 such over flow list.
- 1 11. (Presently Amended) A computer system as defined in claim 1 wherein the
2 status-word fields, when taken together, form a status word that fits in a memory
3 location accessible in a single machine instruction.
- 1 12. (Original) A computer system as defined in claim 11 wherein the parallel-
2 execution operation is a garbage-collection operation.

1 13. (Original) A computer system as defined in claim 11 wherein each status-word
2 field is a single-bit field.

1 14. (Presently Amended) A computer system as defined in claim 13 wherein the
2 ~~activity-indicating value is~~ each single-bit field contains a logic one to indicate
3 that the associated thread is active and the ~~inactivity-indicating value is~~ contains
4 a logic zero to indicate that the associated thread is inactive.

1 15. (Presently Amended) For employing a plurality of threads of execution to perform
2 a parallel-execution operation in which the threads identify tasks dynamically, a
3 method comprising:

4 A) ~~providing a global status word that includes~~ associating a separate
5 status-word field ~~associated with each of the threads; and~~

6 B) ~~so~~ operating the threads in a manner that each thread:

7 i) executes a task-finding routine to find tasks previously
8 identified dynamically and performs tasks thereby found, with the its
9 associated status-word field ~~associated with that thread~~ containing an
10 ~~activity-representing a value~~ indicating it is active, until the task-finding
11 routine finds no more tasks;

12 ii) when the task-finding routine finds no more tasks, sets the
13 contents of the its associated status-word field ~~associated with that thread~~
14 ~~to an inactivity-indicating a value~~ indicating it is inactive;

15 iii) while the status-word field associated with any other thread
16 contains an ~~activity-indicating a value~~ indicating that the other thread is
17 active, searches for a task and, and, if it finds one, sets the its associated
18 status-word field contents to the ~~activity-indicating a value~~ indicating that it
19 is active before attempting to execute a task; and

20 iv) if none of the status-word fields associated with other
21 threads contains an ~~activity-indicating a value~~ indicating that an
22 associated thread is active, terminates its performance of the parallel-
23 execution operation.

- 1 16. (Original) A method as defined in claim 15 wherein the parallel-execution
2 operation is a garbage-collection operation.
- 1 17. (Original) A method as defined in claim 15 wherein:
2 A) each thread has associated with it a respective work queue in
3 which it places task identifiers of tasks that identifies dynamically;
4 B) the task-finding routine executed by an executing thread includes
5 performing an initial search for a task identifiers in the work queue associated
6 with the executing thread and, if that work queue contains no task identifiers that
7 the executing thread can claim, thereafter performing a further search for a task
8 identifier in at least one other task-storage location.
- 1 18. (Original) A method as defined in claim 17 wherein the parallel-execution
2 operation is a garbage-collection operation.
- 1 19. (Original) A method as defined in claim 17 wherein the at least one other task-
2 storage location includes at least one work queue associated with a thread other
3 than the executing thread.
- 1 20. (Original) A method as defined in claim 19 wherein:
2 A) there is a size limit associated with each work queue;
3 B) when a given thread dynamically identifies a given task that would
4 cause the number of task entries in the work queue associated with the given
5 thread to exceed the size limit if a task identifier that identifies it were placed in
6 that work queue, the given thread instead places that task identifier in an
7 overflow list instead of in that work queue; and
8 C) the at least one other task-storage location includes at least one
9 such over-flow list.

- 1 21. (Original) A method as defined in claim 19 wherein the task-finding routine
2 includes selecting in a random manner the at least one work queue associated
3 with a thread other than the executing thread.
- 1 22. (Original) A method as defined in claim 19 wherein the further search includes
2 repeatedly searching a work queue associated with a thread other than the
3 executing thread until the executing thread thereby finds a task or has performed
4 a number of repetitions equal to a repetition limit greater than one.
- 1 23. (Original) A method as defined in claim 22 wherein the task-finding routine
2 includes selecting in a random manner the at least one work queue associated
3 with a thread other than the executing thread.
- 1 24. (Original) A method as defined in claim 17 wherein:
2 A) there is a size limit associated with each work queue;
3 B) when a given thread dynamically identifies a given task that would
4 cause the number of task entries in the work queue associated with the given
5 thread to exceed the size limit if a task identifier that identifies it were placed in
6 that work queue, the given thread instead places that task identifier in an
7 overflow list instead of in that work queue; and
8 C) the at least one other task-storage location includes at least one
9 such over-flow list.
- 1 25. (Presently Amended) A method as defined in claim 15 wherein the status-word
2 fields, when taken together, form a status word that fits in a memory location
3 accessible in a single machine instruction.
- 1 26. (Original) A method as defined in claim 25 wherein the parallel-execution
2 operation is a garbage-collection operation.

1 27. (Original) A method as defined in claim 25 wherein each status-word field is a
2 single-bit field.

1 28. (Presently Amended) A method as defined in claim 27 wherein ~~the activity-~~
2 ~~indicating value is~~ each single-bit field contains a logic one to indicate that the
3 associated thread is active and ~~the inactivity-indicating value is~~ contains a logic
4 zero to indicate that the associated thread is inactive.

1 29. (Presently Amended) A storage medium containing instructions readable by a
2 computer system to configure the computer system to employ a plurality of
3 threads of execution to perform a parallel-execution operation in which the
4 threads identify tasks dynamically and in which the computer system comprises:

5 A) ~~provides a global status word that includes a mechanism that~~
6 associates a separate status-word field ~~associated~~ with each of the threads; and

7 B) ~~so~~ a mechanism that operates the threads in a manner that each
8 thread:

9 i) executes a task-finding routine to find tasks previously
10 identified dynamically and performs tasks thereby found, with ~~the~~ its
11 associated status-word field ~~associated with that thread~~ containing an
12 ~~activity representing a value~~ indicating it is active, until the task-finding
13 routine finds no more tasks;

14 ii) when the task-finding routine finds no more tasks, sets the
15 contents of the its associated status-word field ~~associated with that thread~~
16 ~~to an inactivity-indicating a value~~ indicating it is inactive;

17 iii) while the status-word field associated with any other thread
18 contains ~~an activity-indicating a value~~ indicating that the other thread is
19 active, searches for a task ~~and, and,~~ if it finds one, sets ~~the~~ its associated
20 status-word field contents to ~~the activity-indicating a value~~ indicating that it
21 is active before attempting to execute a task; and

22 iv) if none of the status-word fields associated with other
23 threads contains ~~an activity-indicating a value~~ indicating that an

24 associated thread is active, terminates its performance of the parallel-
25 execution operation.

1 30. (Original) A storage medium as defined in claim 29 wherein the parallel-
2 execution operation is a garbage-collection operation.

1 31. (Original) A storage medium as defined in claim 29 wherein:
2 A) each thread has associated with it a respective work queue in
3 which it places task identifiers of tasks that identifies dynamically;
4 B) the task-finding routine executed by an executing thread includes
5 performing an initial search for a task identifiers in the work queue associated
6 with the executing thread and, if that work queue contains no task identifiers that
7 the executing thread can claim, thereafter performing a further search for a task
8 identifier in at least one other task-storage location.

1 32. (Original) A storage medium as defined in claim 31 wherein the parallel-
2 execution operation is a garbage-collection operation.

1 33. (Original) A storage medium as defined in claim 31 wherein the at least one other
2 task-storage location includes at least one work queue associated with a thread
3 other than the executing thread.

1 34. (Original) A storage medium as defined in claim 33 wherein:
2 A) there is a size limit associated with each work queue;
3 B) when a given thread dynamically identifies a given task that would
4 cause the number of task entries in the work queue associated with the given
5 thread to exceed the size limit if a task identifier that identifies it were placed in
6 that work queue, the given thread instead places that task identifier in an
7 overflow list instead of in that work queue; and
8 C) the at least one other task-storage location includes at least one
9 such over flow list.

- 1 35. (Original) A storage medium as defined in claim 33 wherein the task-finding
2 routine includes selecting in a random manner the at least one work queue
3 associated with a thread other than the executing thread.
- 1 36. (Original) A storage medium as defined in claim 33 wherein the further search
2 includes repeatedly searching a work queue associated with a thread other than
3 the executing thread until the executing thread thereby finds a task or has
4 performed a number of repetitions equal to a repetition limit greater than one.
- 1 37. (Original) A storage medium as defined in claim 36 wherein the task-finding
2 routine includes selecting in a random manner the at least one work queue
3 associated with a thread other than the executing thread.
- 1 38. (Original) A storage medium as defined in claim 31 wherein:
2 A) there is a size limit associated with each work queue;
3 B) when a given thread dynamically identifies a given task that would
4 cause the number of task entries in the work queue associated with the given
5 thread to exceed the size limit if a task identifier that identifies it were placed in
6 that work queue, the given thread instead places that task identifier in an
7 overflow list instead of in that work queue; and
8 C) the at least one other task-storage location includes at least one
9 such over flow list.
- 1 39. (Presently Amended) A storage medium as defined in claim 29 wherein the
2 status-word fields, when taken together, form a status word that fits in a memory
3 location accessible in a single machine instruction.
- 1 40. (Original) A storage medium as defined in claim 39 wherein the parallel-
2 execution operation is a garbage-collection operation.

1 41. (Original) A storage medium as defined in claim 39 wherein each status-word
2 field is a single-bit field.

1 42. (Presently Amended) A storage medium as defined in claim 41 wherein ~~the~~ each
2 single-bit field contains activity-indicating value is a logic one to indicate that the
3 associated thread is active and ~~the inactivity-indicating value is~~ contains a logic
4 zero to indicate that the associated thread is inactive.

1 43. (Presently Amended) A computer signal representing a sequence of instructions
2 that, when executed by a computer system, configures the computer system to
3 employ a plurality of threads of execution to perform a parallel-execution
4 operation in which the threads identify tasks dynamically and in which the
5 computer system comprises:

6 A) ~~provides a global status word that includes~~ a mechanism that
7 associates a separate status-word field ~~associated with each of the threads; and~~

8 B) ~~so~~ a mechanism that operates the threads in a manner that each
9 thread:

10 i) executes a task-finding routine to find tasks previously
11 identified dynamically and performs tasks thereby found, with ~~the~~ its
12 associated status-word field ~~associated with that thread~~ containing an
13 ~~activity-representing~~ a value indicating it is active, until the task-finding
14 routine finds no more tasks;

15 ii) when the task-finding routine finds no more tasks, sets the
16 contents of ~~the~~ its associated status-word field ~~associated with that thread~~
17 ~~to an inactivity-indicating~~ a value indicating it is inactive;

18 iii) while the status-word field associated with any other thread
19 contains ~~an activity-indicating~~ a value indicating that the associated thread
20 is active, searches for a task ~~and, and~~, if it finds one, sets ~~the~~ its
21 associated status-word field contents ~~to the activity-indicating~~ a value
22 indicating that it is active before attempting to execute a task; and

23 iv) if none of the status-word fields contains ~~an activity~~
24 indicating a value indicating that an associated thread is active, terminates
25 its performance of the parallel-execution operation.

1 44. (Original) A computer signal as defined in claim 43 wherein the parallel-execution
2 operation is a garbage-collection operation.

1 45. (Original) A computer signal as defined in claim 43 wherein:
2 A) each thread has associated with it a respective work queue in
3 which it places task identifiers of tasks that identifies dynamically;
4 B) the task-finding routine executed by an executing thread includes
5 performing an initial search for a task identifiers in the work queue associated
6 with the executing thread and, if that work queue contains no task identifiers that
7 the executing thread can claim, thereafter performing a further search for a task
8 identifier in at least one other task-storage location.

1 46. (Original) A computer signal as defined in claim 45 wherein the parallel-execution
2 operation is a garbage-collection operation.

1 47. (Original) A computer signal as defined in claim 45 wherein the at least one other
2 task-storage location includes at least one work queue associated with a thread
3 other than the executing thread.

1 48. (Original) A computer signal as defined in claim 47 wherein:
2 A) there is a size limit associated with each work queue;
3 B) when a given thread dynamically identifies a given task that would
4 cause the number of task entries in the work queue associated with the given
5 thread to exceed the size limit if a task identifier that identities it were placed in
6 that work queue, the given thread instead places that task identifier in an
7 overflow list instead of in that work queue; and

8 C) the at least one other task-storage location includes at least one
9 such over flow list.

1 49. (Original) A computer signal as defined in claim 47 wherein the task-finding
2 routine includes selecting in a random manner the at least one work queue
3 associated with a thread other than the executing thread.

1 50. (Original) A computer signal as defined in claim 47 wherein the further search
2 includes repeatedly searching a work queue associated with a thread other than
3 the executing thread until the executing thread thereby finds a task or has
4 performed a number of repetitions equal to a repetition limit greater than one.

1 51. (Original) A computer signal as defined in claim 50 wherein the task-finding
2 routine includes selecting in a random manner the at least one work queue
3 associated with a thread other than the executing thread.

1 52. (Original) A computer signal as defined in claim 45 wherein:
2 A) there is a size limit associated with each work queue;
3 B) when a given thread dynamically identifies a given task that would
4 cause the number of task entries in the word queue associated with the given
5 thread to exceed the size limit if a task identifier that identifies it were placed in
6 that work queue, the given thread instead places that task identifier in an
7 overflow list instead of in that work queue; and
8 C) the at least one other task-storage location includes at least one
9 such over flow list.

1 53. (Presently Amended) A computer signal as defined in claim 43 wherein the
2 status-word fields, taken together, form a status word that fits in a memory
3 location accessible in a single machine instruction.

1 54. (Original) A computer signal as defined in claim 53 wherein the parallel-execution
2 operation is a garbage-collection operation.

1 55. (Original) A computer signal as defined in claim 53 wherein each status-word
2 field is a single-bit field.

1 56. (Presently Amended) A computer signal as defined in claim 55 wherein the
2 ~~activity-indicating value is~~ each single-bit field contains a logic one to indicate
3 that the associated thread is active and contains the inactivity-indicating value is
4 a logic zero to indicate that the associated thread is inactive.

1 57. (Presently Amended) A computer system that employs a plurality of threads of
2 execution to perform a parallel-execution operation in which the threads identify
3 tasks dynamically, the computer system including:

4 A) means for ~~providing a global status word that includes~~ associating a
5 separate status-word field ~~associated~~ with each of the threads; and

6 B) means for ~~so~~ operating the threads in a manner that each thread:

7 i) executes a task-finding routine to find tasks previously
8 identified dynamically and performs tasks thereby found, with ~~the~~ its
9 associated status-word field ~~associated with that thread~~ containing an
10 ~~activity-representing a value~~ indicating it is active, until the task-finding
11 routine finds no more tasks;

12 ii) when the task-finding routine finds no more tasks, sets the
13 contents of the its associated status-word field ~~associated with that thread~~
14 ~~to an inactivity-indicating a value~~ indicating it is inactive;

15 iii) while the status-word field associated with any other thread
16 contains ~~an activity-indicating a value~~ indicating that the other thread is
17 active, searches for a task ~~and, and~~, if it finds one, sets the status-word
18 field contents to the activity-indicating a value indicating that it is active
19 before attempting to execute a task; and

20 iv) if none of the status-word fields associated with other
21 threads contains an ~~activity indicating~~ a value indicating that an
22 associated thread is active, terminates its performance of the parallel-
23 execution operation.